AMENDMENTS

In the Claims

1. (Currently Amended) A method for initializing a memory page, the method comprising:

in response to a memory operation by a first thread, allocating a memory page;

generating a request for a second thread to initialize the allocated memory page; and

initializing the allocated memory page by the second thread in accordance with the request; and,

indicating the memory page as being in a pager input/output state after allocating the memory page; and

indicating the allocated memory page as being in a normal state after initializing the memory page.

2. (Original) The method of claim 1 further comprising:

putting the first thread into a sleep state prior to initialization of the allocated memory page; and

in response to completion of initialization of the allocated memory page, putting the first thread into a runnable state.

3. (Original) The method of claim 1 further comprising:

zeroing the allocated memory page to initialize the allocated memory page.

4. (Original) The method of claim 1 further comprising:

copying contents from a source page to the allocated memory page to initialize the allocated memory page, wherein the request identifies the source page.

5. (Original) The method of claim 1 further comprising:

receiving an interrupt prior to allocating the memory page; and returning from the interrupt after generating the request for the second thread to initialize the allocated memory page.

6. (Original) The method of claim 5 further comprising:
identifying the interrupt as a result of a page fault.

7. (Original) The method of claim 5 further comprising:
identifying the interrupt as a result of a copy-on-write fault.

8. (Original) The method of claim 1 wherein the memory page is allocated to an application comprising the first thread.

9. (Original) The method of claim 1 wherein the second thread is a kernel worker thread.

10. (Canceled)

11. (Currently Amended) A computer program product on a computer readable medium for use in a data processing system for initializing a memory page, the computer program product comprising:

means for allocating a memory page in response to a memory operation by a first thread;

means for generating a request for a second thread to initialize the allocated memory page; and

means for initializing the allocated memory page by the second thread in accordance with the request; and

means for indicating the memory page as being in a pager input/output state after allocating the memory page; and

means for indicating the allocated memory page as being in a normal state after initializing the memory page.

12. (Original) The computer program product of claim 11 further comprising:

means for putting the first thread into a sleep state prior to initialization of the allocated memory page; and

means for putting the first thread into a runnable state in response to completion of initialization of the allocated memory page.

13. (Original) The computer program product of claim 11 further comprising:

means for zeroing the allocated memory page to initialize the allocated memory page.

14. (Original) The computer program product of claim 11 further comprising:

means for copying contents from a source page to the allocated memory page to initialize the allocated memory page, wherein the request identifies the source page.

15. (Original) The computer program product of claim 11 further comprising:

means for receiving an interrupt prior to allocating the memory page; and

means for returning from the interrupt after generating the request for the second thread to initialize the allocated memory page.

16. (Original) The computer program product of claim 15 further comprising:

means for identifying the interrupt as a result of a page fault.

17. (Original) The computer program product of claim 15 further comprising:

means for identifying the interrupt as a result of a copy-on-write fault.

18. (Original) The computer program product of claim 11 wherein the memory page is allocated to an application comprising the first thread.

19. (Original) The computer program product of claim 11 wherein the second thread is a kernel worker thread.

20. (Canceled)

21. (Original) An apparatus for initializing a memory page, the apparatus

comprising:

means for allocating a memory page in response to a memory operation by a first thread;

means for generating a request for a second thread to initialize the allocated memory page; and

means for initializing the allocated memory page by the second thread in accordance with the request; and,

means for indicating the memory page as being in a pager input/output state after allocating the memory page; and

means for indicating the allocated memory page as being in a normal state after initializing the memory page.

22. (Original) The apparatus of claim 21 further comprising:

means for putting the first thread into a sleep state prior to initialization of the allocated memory page; and

means for putting the first thread into a runnable state in response to completion of initialization of the allocated memory page.

23. (Original) The apparatus of claim 21 further comprising:

means for zeroing the allocated memory page to initialize the allocated memory page.

24. (Original) The apparatus of claim 21 further comprising:

means for copying contents from a source page to the allocated memory page to initialize the allocated memory page, wherein the request identifies the source page.

25. (Original) The apparatus of claim 21 further comprising:

means for receiving an interrupt prior to allocating the memory page; and

means for returning from the interrupt after generating the request for the second thread to initialize the allocated memory page.

26. (Original) The apparatus of claim 25 further comprising:

means for identifying the interrupt as a result of a page fault.

27. (Original) The apparatus of claim 25 further comprising:
means for identifying the interrupt as a result of a copy-on-write fault.

28. (Original) The apparatus of claim 21 wherein the memory page is allocated to an application comprising the first thread.

29. (Original) The apparatus of claim 21 wherein the second thread is a kernel worker thread.

30. (Canceled)